IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

APPEAL NO:

In Re Application of: Balaji et al.

Confirmation No.: 8889

Serial No.: 10/620,581

Filed: July 15, 2003

For: FLEXIBLE ARCHITECTURE COMPONENT (FAC) FOR EFFICIENT DATA INTEGRATION AND INFORMATION INTERCHANGE USING WEB SERVICES

APPEAL BRIEF

Stephen G. Sullivan
Attorney for Appellants
Strategic Patent Group, PC

TOPICAL INDEX

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In Re Application of: Balaji et al.	Confirmation No: 8889
Serial No: 10/620,581	Group Art Unit: 2194
Filed: July 15, 2003	Examiner: Price, Nathan E.

For: FLEXIBLE ARCHITECTURE COMPONENT (FAC) FOR EFFICIENT DATA INTEGRATION AND INFORMATION INTERCHANGE USING WEB SERVICES

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

Appellants herein file an Appeal Brief drafted in accordance with the provisions of 37 C.F.R. §41.37 as follows:

I REAL PARTY IN INTEREST

Appellants respectfully submit that the above-captioned application is assigned, in its entirety to LSI Corporation of Milpitas, CA.

II RELATED APPEALS AND INTERFERENCES

Appellants state that, upon information and belief, he is not aware of any co-pending appeal or interference which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III STATUS OF CLAIMS

Application Serial No. 10/620,581 (the instant application), as originally filed, included claims 1-30 Claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33 are presently pending. In response to an Office Action dated January 22, 2007, claims 1-2, 6, 9, 13-14, 18, 21, and 25-26 were amended and claims 4-5, 8, 10, 16-17, 20, and 22 were cancelled. In response to a Final Office Action dated April 18, 2007, claims 1, 3, 6-7, 13, 15, 18-19, 25, 27, and 29 were amended. In a Supplemental Amendment filed on September 26, 2007, claims 31-33 were added, claims 2-3, 6-7, 14, 18-19, and 26 were amended, and claims 1, 11-13, 23-25, and 29-30 were canceled. Claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33 are on appeal and all applied prospective rejections concerning claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33 are being appealed herein.

IV STATUS OF AMENDMENTS

Amendments submitted in the Supplemental Amendment filed on September 26, 2007 were entered by the Examiner.

V SUMMARY OF CLAIMED SUBJECT MATTER

The present invention addresses the situation where multiple client applications need to access respective data sources that are stored in different formats and are not directly accessible by the other client applications. Accessibility is provided by storing data from the respective data sources in a database at a server in a manner where the data is standardized. The invention provides this standardized data by providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format, and modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in XML files. Each of the XML files is then submitted to an import repository at the server. Each of the XML files in the import repository are validated against a document type definition (DTD) corresponding to the respective data sources. The validated XML files are parsed, and name/value pairs are stored in the database at the server according to a hierarchy specified by the corresponding DTD. Thus, the XML files are collected and validated in the import repository before being stored in the database.

The collecting of the XML files in the import repository provides the following advantages: 1) it provides a separate collection of XML files to ensure that data sources can be completely validated and recorded before entering the database; 2) a separate staging area keeps the XML files in isolation of the operational database, and hence minimizes the impact on the integrated data warehouse (if things go wrong); and a separate staging area for a client applications provides an ability to track transactions

independent of the database. (See specification at p. 17, lines 14-23) Because the XML files are provided and validated in the import repository in the claimed manner, prior to being stored in the database, true standardization of the data from the data sources is provided.

Independent claim 31 recites a method for providing data integration and exchange between a plurality of client applications over a network, wherein each of the client applications access a respective data source. The method comprises:

(a) providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format, wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications (page 9, line 23, to page 10, line 2; and page 10, lines 18-21);

(b) modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in XML files (page 10, lines 10-14; and page 11, lines 1-4);

(c) submitting each of the XML files to an import repository at a server (page 17, lines 9-12);

(d) validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources (page 18, lines 12-14); and

(e) parsing the validated XML files in the import repository and storing name/value pairs in a database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications (page 18, lines 14-17).

Independent claim 32 recites computer-readable medium containing program instructions for providing data integration and exchange between a plurality of client applications over a network, wherein each of the client applications access a respective data source. The program instructions are for:

(a) providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format, wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications (page 9, line 23, to page 10, line 2; and page 10, lines 18-21);

(b) modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in an XML file (page 10, lines 10-14; and page 11, lines 1-4);

(c) submitting each of the XML files from the client applications to an import repository at a server (page 17, lines 9-12);

(d) validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources (page 18, lines 12-14); and

(e) parsing the validated XML files in the import repository and storing name/value pairs in a database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications (page 18, lines 14-17).

Independent claim 33 recites a data integration system that comprises:

a network (Figure 1, element 14; and page 9, line 20);

a server coupled to the network, the server including a schema registry, an import repository, an XML loader, a database, and a published adapter API at each of the client applications that provides a first set of methods for translating data in respective data source into XML format (Figure 1, element 16; and page 10, lines 4-21); and

a plurality of client applications coupled to the network and in communication with the server, wherein each of the client applications access the respective data source, and wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications (Figure 1, elements 12, and page 9, line 19, to page 10, line 2), and

wherein at least a portion of the client applications includes a corresponding schema definition and document type definition (DTD) registered with the schema registry, and the portion of the client applications includes generation logic for making calls to the first set of methods in the adapter API, such that data in the respective data sources are converted into XML format according to the corresponding schema definition and stored in XML files (page 10, lines 10-14; and page 11, lines 1-4),

wherein each of the XML files is submitted to the import repository at the server (page 17, lines 9-12), wherein each of the XML files in the import repository is validated against the corresponding DTD, and wherein the XML loader parses the validated XML files in the import reposition and stores name/value pairs in the database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications (page 18, lines 12-17).

VI GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33 stand rejected under 35 U.S.C. §103(a) as being over U.S. Publication No. 2003/0204645 issued to Sharma in view of U.S. Patent No. 6,948,174 issued to Chiang and in view of U.S. Patent No. 6,880,125 issued to Fry.

VII ARGUMENTS

1. Rejection of claims under 35 U.S.C. §103(a) as being over U.S. Publication No. 2003/0204645 issued to Sharma in view of U.S. Patent No. 6,948,174 issued to Chiang and U.S. Patent No. 6,880,125 issued to Fry

Appellants respectfully submit that the rejections under 35 U.S.C. § 103(a) are not supported because the Examiner has failed to clearly articulate why Sharma in view of Chiang and Fry teaches or suggests the method, computer-readable medium, and system recited in the claims.

i) Claim 31 is not unpatentable over Sharma in view of Chiang and Fry

In contrast to the present invention, Sharma is directed to providing a servlet container-based Web service endpoint in a remote procedure call-based distributed computing system [003]. Sharma describes a computing system which provides a service associated with a service endpoint class that implements a service endpoint interface by packaging the service endpoint class and interface in an archive file. The computing system may define a service endpoint based on information included in the archive file and modify the archive file with information associated with the service

endpoint definition. The modified archive file may then be deployed on a servlet container operating in the computing system. [0011]. The computing system may create a Web Services Description Language (WSDL) document that describes the service endpoint based on the information contained in the modified archive file and export the WSDL document such that a remote computing system may use the WSDL document to access the service endpoint. [0012].

As is known in the art, a remote procedure call (RPC) is a technology that allows a computer program to cause a subroutine or procedure to execute in another address space, typically on another computer on a shared network, without the programmer explicitly coding the details for this remote interaction. That is, the programmer would write essentially the same code whether the subroutine is local to the executing program, or remote. RPC Web services present a distributed function (or method) call interface that is sometimes implemented by mapping services directly to language-specific functions or method calls (Wikipedia).

By contrast, the present invention has nothing whatsoever to do with RPC, whereby the client is invoking procedures or methods on the server. Instead, the claims of the present of invention are directed to, inter alia, client-side adapter APIs.

Sharma Fails to Teach or Suggest the Elements of Claim 31

Applicants agree with the Examiner that Sharma fails to specifically disclose saving XML format data from respective data sources in XML files, as partially recited in step (b) of claim 31, and fails to specifically disclose "validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the

respective data sources," as recited in step (d) of claim 31. However, applicants would go further, and submit that Sharma fails to teach or suggest the remaining elements of claim 31 as well.

First, Sharma fails to teach or suggest "providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format," as recited in step (a).

The Examiner cited FIG. 5, paragraphs 5, 8-10, 37, 115, and 127 of Sharma for teaching this step. However, paragraphs 5 and 6 merely describe a generic client/server computing environment and provide the definition of remote procedure calls (RPCs), while paragraphs 8-10 and 37 discuss XML and XML based RPC services and environments. Paragraph 127 describes that the APIs may be implemented to support an extensible type mapping between XML data types and Java types. Although paragraph 115 describes that a client may use client-side APIs, which presumably the Examiner considers analogous to the claimed "adapter API", Sharma's client-side APIs use a stub 515 to invoke a remote method on service endpoint 555 *in a server*. Sharma does not teach or suggest in this passage, or any other that Applicants could find, where client-side APIs "translate data in the respective data sources into XML format," as claimed.

In the Final Office Action dated April 14, 2008, the Examiner again referred to paragraph 127 of Sharma as disclosing that APIs that perform mapping for XML can be in the client. However, paragraph 127 of Sharma merely states that the "APIs may enable server 110 and client 130 to develop pluggable serializers and deserializers to support extensible mapping between any Java type and XML data type." This still does

not teach or even suggest that client-side API's "translate data in the respective data sources into XML format," as claimed.

Moreover, Applicants could find no teaching or suggestion in Sharma of a plurality of client applications, "wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications," as further recited in step (a). Sharma may teach that his system may include a plurality of clients [065], and that the clients may require data stored another computing node, typically a server [005], but Sharma falls short in teaching that the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications.

In the Final Office Action dated April 14, 2008, the Examiner again referred to paragraph 37 of Sharma as disclosing this feature. However, this paragraph merely mentions that "a Java based service client may be capable of using an XML-based RPC service deployed in a non-Java based platform." There is no mention or suggestion that data sources are "not directly accessible by the other client applications," as claimed.

Because Sharma fails to teach or suggest "an adapter API at each of the client applications that provides a first set of methods...," in step (a) as argued above, it follows that Sharma cannot teach or suggest "modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition," as recited in step (b).

In the Final Office Action dated April 14, 2008, the Examiner again referred to paragraphs 72, 115, 127 and 130 of Sharma as disclosing the modifying feature. However, paragraph 72 merely describes holder classes that may be used by a mapping tool. As describe above, paragraph 115 merely describes that client-side APIs use a stub 515 to invoke a remote method on service endpoint 555 in a server. Paragraph 127 merely describes extensible type mapping, and paragraph 130 merely describes a serializer interface and a deserializer interface. None of these paragraphs mention or suggest "modifying each of the client applications" and, specifically, "modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition," as recited in step (b).

The Examiner cited paragraphs 7, 72, 91, 115, 127, and 130 of Sharma for teaching this step. Paragraph 115 may describe that the client-side APIs uses stub 515 to invoke a remote method on service endpoint 555. However, the remote method invoked is on the server, rather than a method of the client API. In addition, the remote method invoked in Sharma fails to "convert data in respective data sources into XML format according to a registered schema definition," as claimed.

Sharma also fails to teach or suggest "submitting each of the XML files to an *import repository* at a server", as recited in step (c), in combination with "parsing the validated XML files in the import repository and storing name/value pairs in *a database* at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications," as recited in step (e).

The Examiner cites paragraphs 5, 115, and 127 for submitting the XML files to an import registry at the server. However, none of these passages describe a process by which XML files created by client-side APIs are submitted to an import registry at the server. Sharma also fails to teach a database of the server that is separate from the import registry. In fact, an electronic search of Sharma for the terms "import registry" and "database" revealed no matches. Because Sharma fails to teach a database apart from an import registry, Sharma cannot teach or suggest "parsing the validated XML files in the import repository and storing name/value pairs in *a database* at the server according to a hierarchy specified by the corresponding DTD."

Secondary References Fail to Cure Sharma

A secondary reference stands or falls with the primary reference. Because Sharma fails to teach or suggest the steps described above, a combination of Sharma, Chiang, and Fry also fails to teach or suggest the claimed invention. Accordingly, claim 31 is patentable over the references for at least this reason.

As stated above, the Examiner admits that Sharma fails to specifically disclose saving XML format data from the respective data sources in XML files. The Examiner cited Chiang for teaching saving the XML format data an XML file.

Chiang discloses using the XML file to convert languages between a user application and a server. Although Chiang discloses the use of an XML file, Chiang is concerned with a user application and a server being able to communicate, not the standardization of data from multiple data sources in different formats at a database. Chiang does not teach or suggest how to collect, validate, parse, or store XML files

from multiple data sources in different formats, such that the standardization of data from the data sources is provided across user applications. Even assuming that multiple user applications can communicate with the server as disclosed by Chiang, this only teaches how each individual user application can communicate with the same server. This is not a teaching of how the data from these multiple user applications are truly standardized for access by each other.

Sharma in view of Chiang thus fails to teach or suggest modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in XML files, submitting each of the XML files to an import repository at a server, validating each of the XML files in the import repository against a DTD corresponding to the respective data sources, and parsing the validated XML files in the import repository and storing name/value pairs in a database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources, as recited in amended independent claims 31, 32, and 33.

The Examiner cited Fry for teaching validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources. However, Fry likewise fails to provide any teaching regarding the standardization of data from multiple data sources in different formats via client-side adapter APIs and the claimed process. Therefore, a combination of Sharma, Chiang, and Fry also fails to teach or suggest independent claim 31, and this claim is allowable over Sharma in view of Chiang and Fry.

ii) Claims 32 and 33 are not unpatentable over Sharma in view of Chiang and Fry

Independent claims 32 and 33 include the same or similar limitations as claim 31. Therefore, claims 32 and 33 are allowable for at least the same reasons as claim 31.

iii) Claims 2-3, 6-7, 9, 14-15, 18-19, 21, and 26-28 are not unpatentable over Sharma in view of Chiang and Fry

Dependent claims 2-3, 6-7, 9, 14-15, 18-19, 21, and 26-28 depend from independent claims 31, 32, and 33, respectively. Accordingly, the above-articulated arguments related to independent claims 31, 32, and 33 apply with equal force to claims 2-3, 6-7, 9, 14-15, 18-19, 21, and 26-28, which are thus allowable over the cited references for at least the same reasons as claims 31, 32, and 33.

For the reasons set forth above, it is respectfully submitted that the section 103(a) rejection of claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33 based on Sharma in view of Chiang and Fry has been overcome.

Accordingly, Appellants respectfully request withdrawal of the rejection under 35 U.S.C. 103(a) respectfully request that the Board reverse the final rejection of claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33.

For all the foregoing reasons, it is respectfully submitted that claims 2-3, 6-7, 9, 14-15, 18-19, 21, 26-28, and 31-33 (all the claims presently in the application) are

patentable. Thus, Appellants respectfully request that the Board reverse the rejection of all the appealed claims and find each of these claims allowable.

Note: For convenience of detachment without disturbing the integrity of the remainder of pages of this Appeal Brief, Appellants' "APPENDIX" sections are contained on separate sheets following the signatory portion of this Appeal Brief.

Respectfully submitted,

Strategic Patent Group, PC

August 7, 2008
Date

/Stephen G. Sullivan/
Stephen G. Sullivan
Attorney for Appellant(s)
Reg. No. 38,329
(650) 969-7474

VIII CLAIMS APPENDIX

1 (Canceled)

2 (Previously Presented) The method of claim 31 further including:

(f) including a second set of methods in the adapter API for the client applications that provides consumption logic and methods for automatically exporting data defined in a Web-based schema registry, from the database into the client applications using Web services.

3 (Previously Presented) The method of claim 2 further including: registering the respective data sources with a schema registry in order to create the schema definition and the document type definition (DTD).

4 (Canceled)

5 (Canceled)

6 (Previously Presented) The method of claim 3 wherein the adapter API includes an XML API comprising the first set of methods and the second set of methods, wherein the first set of methods comprises a Writer API, and the second set of methods comprises a Reader API.

7 (Previously Presented) The method of claim 6 wherein the client applications are modified with generator logic that makes calls to methods comprising the adapter API,

wherein once called, the Writer API converts data into the XML format in memory and saves the XML format data in the XML files, which are then transported to the server.

8 (Canceled)

9 (Previously Presented) The method of claim 7 wherein the adapter further includes verification code that verifies the generated XML data against the DTD defined in schema registry.

10 (Canceled)

11 (Canceled)

12 (Canceled)

13 (Canceled)

14 (Previously Presented) The computer-readable medium of claim 32 further including instruction of:

(c) including a second set of methods in the adapter API for the client applications that provides consumption logic and methods for automatically exporting data defined in a Web-based schema registry, from the database into the client applications using Web services.

15 (Previously Presented) The computer-readable medium of claim 14 further including the instruction of: registering the respective data sources with a schema registry in order to create the schema definition and the document type definition (DTD).

16 (Canceled)

17 (Canceled)

18 (Previously Presented) The computer-readable medium of claim 15 wherein the adapter API includes an XML API comprising the first set of methods and the second set of methods, wherein the first set of methods comprises a Writer API, and the second set of methods comprises a Reader API.

19 (Previously Presented) The computer-readable medium of claim 18 wherein the client applications are modified with generator logic that makes calls to methods comprising the adapter API, wherein once called, the Writer API converts data into the XML format in memory and saves the XML format data in the XML files, which are transported to the server.

20 (Canceled)

21 (Previously Presented) The computer-readable medium of claim 19 wherein the adapter further includes verification code that verifies the generated XML data against the DTD defined in schema registry.

22 (Canceled)

23 (Canceled)

24 (Canceled)

25 (Canceled)

26 (Previously Presented) The system of claim 33 wherein the adapter API further includes a second set of methods for the client applications that that provides consumption logic and methods for automatically exporting data defined in the schema registry, from the database into the client applications using Web services.

27 (Previously Presented) The system of claim 26 wherein the adapter API includes an XML API comprising the first set of methods and the second set of methods, wherein the first set of methods comprises a Writer API and the second set of methods comprises a Reader API.

28 (Original) The system of claim 27 wherein the server further includes a schema generator for generating the schema definition, a DTD generator for generating the

DTD, and an adapter software kit that is downloaded from the server and used to incorporate the adapter API into the client applications.

29 (Canceled)

30 (Canceled)

31 (Previously Presented) A method for providing data integration and exchange between a plurality of client applications over a network, wherein each of the client applications access a respective data source, the method comprising:

(a) providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format, wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications;

(b) modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in XML files;

(c) submitting each of the XML files to an import repository at a server;

(d) validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources; and

(e) parsing the validated XML files in the import repository and storing name/value pairs in a database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications.

32 (Previously Presented) A computer-readable medium containing program instructions for providing data integration and exchange between a plurality of client applications over a network, wherein each of the client applications access a respective data source, the program instructions for:

(a) providing an adapter API at each of the client applications that provides a first set of methods for the client applications to use to translate data in the respective data sources into XML format, wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications;

(b) modifying each of the client applications to invoke the first set of methods in the adapter API to convert the data in the respective data sources into XML format according to a registered schema definition and saving the XML format data from the respective data sources in an XML file;

(c) submitting each of the XML files from the client applications to an import repository at a server;

(d) validating each of the XML files in the import repository against a document type definition (DTD) corresponding to the respective data sources; and

(e) parsing the validated XML files in the import repository and storing name/value pairs in a database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications.

33 (Previously Presented) A data integration system, comprising:

a network;

a server coupled to the network, the server including a schema registry, an import repository, an XML loader, a database, and a published adapter API at each of the client applications that provides a first set of methods for translating data in respective data source into XML format; and

a plurality of client applications coupled to the network and in communication with the server, wherein each of the client applications access the respective data source, and wherein the data sources of each of the client applications are stored in different formats and are not directly accessible by the other client applications, and

wherein at least a portion of the client applications includes a corresponding schema definition and document type definition (DTD) registered with the schema registry, and the portion of the client applications includes generation logic for making calls to the first set of methods in the adapter API, such that data in the respective data sources are converted into XML format according to the corresponding schema definition and stored in XML files,

wherein each of the XML files is submitted to the import repository at the server, wherein each of the XML files in the import repository is validated against the corresponding DTD, and wherein the XML loader parses the validated XML files in the import reposition and stores name/value pairs in the database at the server according to a hierarchy specified by the corresponding DTD, thereby standardizing the data from the data sources of the client applications.

IX EVIDENCE APPENDIX

(None)

X RELATED PROCEEDINGS APPENDIX

(None)